Sorbonne Université Pierre et Marie Curie

TELECOM ParisTech

IG3DA
STYLIT : Illumination-Guided Example-Based Stylization of 3D Renderings

Mario Viti
mario.viti@etu.upmc.fr

February 12, 2018

0.1 Stylit style transfer

This algorithm proposes[FJL+16] an approach to example-based stylization of 3D renderings that better preserves the rich expressiveness of hand made artwork. Unlike previous techniques, which are mainly guided by colors and normals, this approach is based on light propagation in the scene. This novel type of guidance can distinguish among context-dependent illumination effects, for which artists typically use different stylization techniques, and delivers a look closer to realistic artwork.

0.1.1 Artistic hand made styles

An artistic hand made style can represent the same object but result in different images, the style can be described by decomposing it into its main factors: *Self Shadow, Cast Shadow, Indirect reflection, Reflection highlights, Color, Texture* are unique expressions of a particular style. Figure 1 examples of stylized still life paintings. Note how unique stylization was used to depict different illumination effects, e.g., pink shading on the bottom of the apple (left) or blue tint in the Kerry shadow region of the red pepper (right).

Figure 1: Image courtesy Daley (left) and Gail Sibley via howtopastel.com (right)

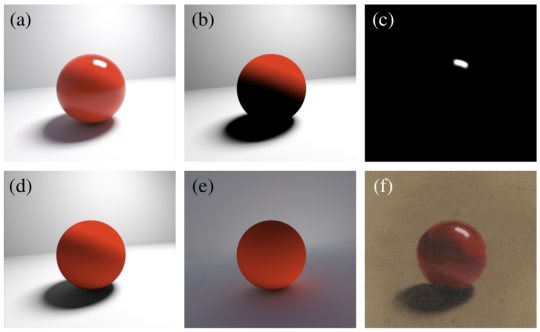


Figure 2: Image LPEs

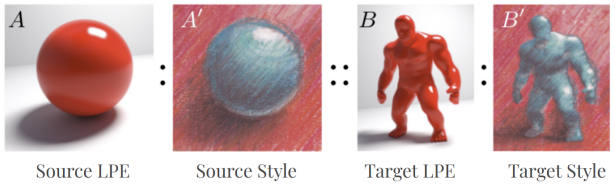
0.1.2 Light Path Expression

Photorealistic rendering can be also be decomposed into terms and factors that contribute to the overall look or render of a scene, terms and factors In light transport, light-surface interactions can generally be classified as either diffuse or specular. Examining the interactions that happen on a path from a light source to the sensor lets us distinguish most important global illumination effects. This technique, known as Light Path Expressions (LPEs).Each LPE can be visualized into separated Maps based on the fundamental property of the rendering equation the physically plausible BRDF (Bidirectional reflectance distribution function), Multiple Bounces diffuse and Global Illumination Models.

Figure 2 An example of a style exemplar with Light Path Expression images: (a) full global illumination render, (b) direct diffuse (LDE), (c) direct specular (LSE), (d) first two diffuse bounces (LD1,2E), (e) diffuse interreflection (L.*DDE), (f) hand-drawn style image. Exemplar image c Daichi Ito.

0.2 Image Analogies

Image analogies to compare this 2 different kind of images and theorise correspondences between synythetic and hand made images of the same subject. A shpere is both represented as a syntethical image composed of LPE channels A and an hand made style A'. The goal of this analogy is to synthesize a hand made styled version of target B only from images A, A' B. We can formulate this problem in optimization terms by defining the energy function E as the weighted sum so square differences between A A' and BB'. The task is to take three multi-channel images A (exemplar scene rendered with LPE channels), A 0 (stylized exemplar aligned to the exemplar scene), and B (target scene rendered with LPE channels), and synthesize a new target image B 0 such that A : A' :: B : B'.



- Analogies are defined by a global energy
  $\mathbf{A} = [A,A']$, $\mathbf{B} = [B,B']$

$$E(\mathbf{A},\mathbf{B},p,q,\mu) = ||A'(p) - B'(q)||^2 + \mu||A(p) - B(q)||^2$$

- The Target result is the image $\mathbf{B}$ that minimizes the global energy

$$\sum_{q\in\mathbf{B}} \min_{p\in\mathbf{A}} E(\mathbf{A},\mathbf{B},p,q,\mu)$$

Figure 3: Image analogies and energy formulation.

0.2.1 E-M like algorithm

First the nearest neighbour in terms of the energy E is found for each patch of B in A and it is than assigned to it in the reconstructed target image B' this iterative approach repeats this 2 steps several times in order to reduce the energy E. Here $B_{k=0}$ denotes the current pixel values stored in B' and B_{k+1} the updated values. NNF is the nearest neighbour field that assigns source patch to each target patch and function Average computes the average color of colocated pixels in neighbour patches.

input : multi-channel images $\mathbf{A} = \{A, A'\}$ and $\mathbf{B}_k = \{B, B'_k\}$
output: synthesized target image B'_{k+1}

for *each pixel* $q \in \boldsymbol{B}_k$ **do**
  $NNF(q) = \underset{p\in\mathbf{A}}{\operatorname{argmin}} E(\mathbf{A}, \mathbf{B}_k, p, q, \mu)$

for *each pixel* $q \in \boldsymbol{B}_k$ **do**
  $B'_{k+1}(q) = Average(\mathbf{A}, NNF, q)$

Figure 4: EM iteration.

0.3 Patch Match

At the core of the stylit application to compute the NNF there's PATCH MATCH[BSFG09]: The insights driving this algorithm are that some good patch matches can be found via random sampling, and that natural coherence in the images allows to propagate such matches quickly to surrounding areas. It is composed of 2 main phases: Propagation and Random Search.

- *Propagation*:first during Propagation each patch is visited in scan order (L-R,T-B) and receives 2 guesses of a better match form its L-T neighbours. Based on these guesses a new match will be chosen as the minimum of the guesses and the current best match. For pair iterations the order is inverted to Bottom right and inverse scan order.
- *Random Search*:After propagation each patch searches randomly over a decreasingly wide area, this step is more expensive and it has a logarithmic complexity in terms of the widest dimension of the image.

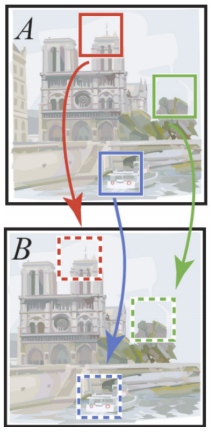


Figure 5: Patch Match task is to find matching patches between images.

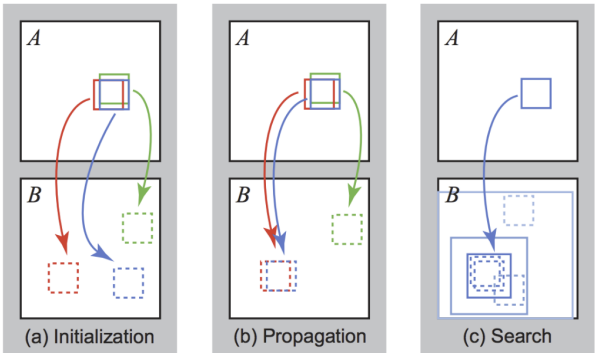
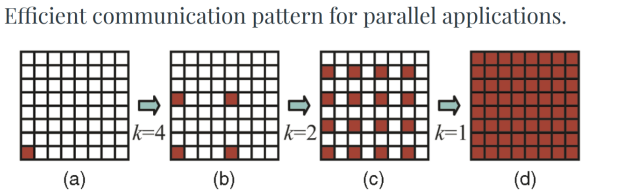


Figure 6: The 2 (plus initailization) phases of the algorithm: Propagation and Random Search.

0.3.1 Jump flooding extension

The propagation step is inheritably a serial operation but it can be parallelized by using the jumping flood communication pattern[RT06]. This algorithm has been developed and shown its efficacy in computing Voronoi diagrams with a Logarithmic number of calls to GPU. The authors of the PatchMatch algorithm hints to this solution however no further detail is available on its implementation. Jumping flood original algorithm is implemented in OpenGL, but a developed a CUDA version is submitted.



Efficient communication pattern for parallel applications.

a) seed position in red, non seed in white.
b) the seed communicates its position in a copy grid, non seed locations reading the communication and updating to nearest seed.
c) d) flooding occupies all the grid.

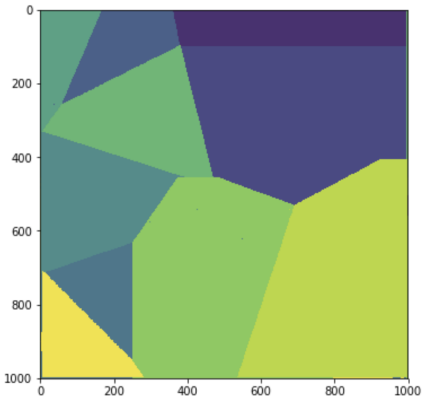


Figure 7: Patch Match task is to find matching patches between images.

0.3.2 Results with Patch Match direct NNF

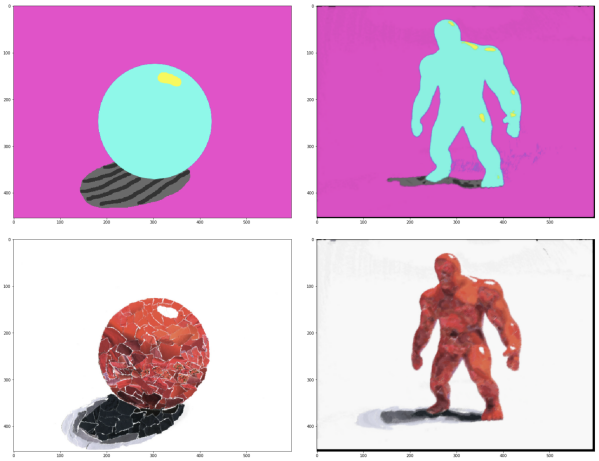


Figure 8: Patch Match application to style transfer TOP: paint noob style, BOT: collage.

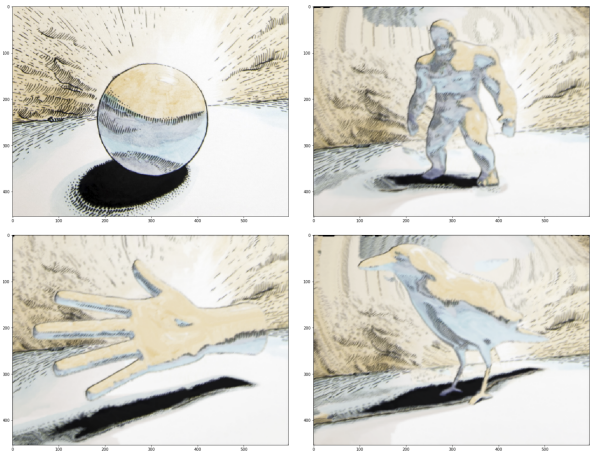


Figure 9: Patch Match direct fails when artistic style is more complex.

0.4 Reversed NNF

[JFA+15] Proposed to inverse the direction of the NNF retrival from the source to the destination. This method however has the inconvenient of collision, multiple source patches will be assigned to the same target patch. The idea is to mask occupied target patches and repeat the retrival until full coverage.

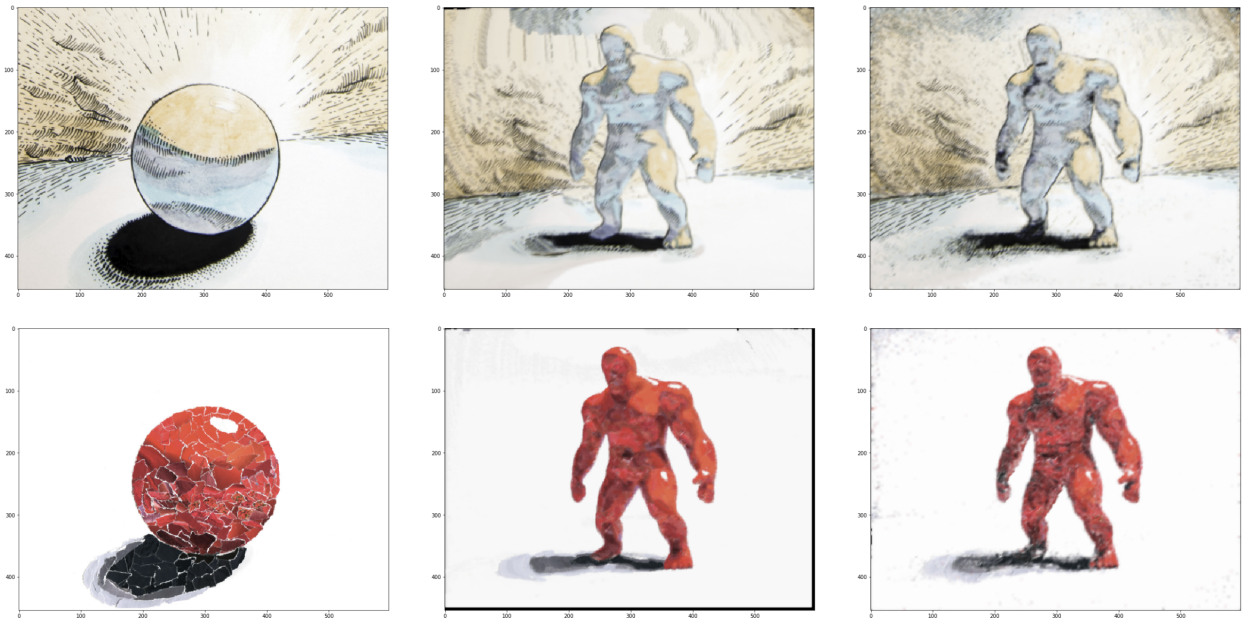


Figure 10: LEFT: artistic style input, CENTER: Patche Match, RIGHT: reversed NNF retrival.

0.4.1 Erroneous assignment filtering

However this method enforces erroneous assignments when all target suitable positions are occupied, by plotting the sorted error for each assignment a hyperbolic profile can be fitted and consequently the

knee point (diamond) can be calculated, all assignments whit an error above this point is unmasked and retrieval continues until the energy diminishes.

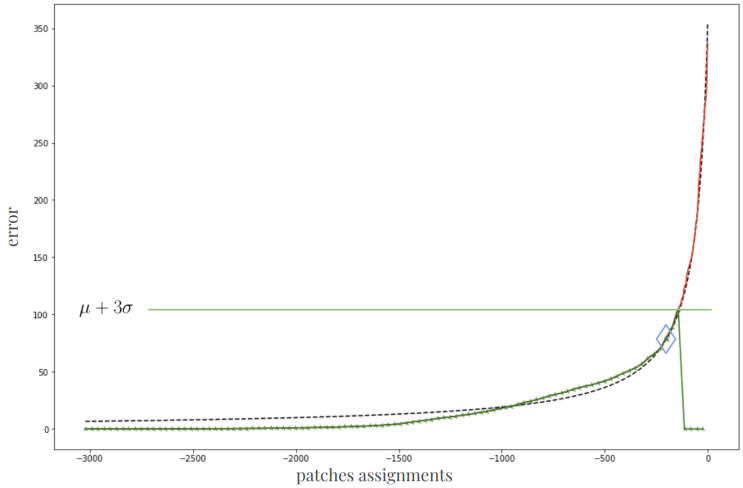


Figure 11: Another strategy is to filter all error above $\mu + 3\sigma$ I've noticed that the knee point calculate a similar threshold so I opted for this strategy as it does not need sorting.

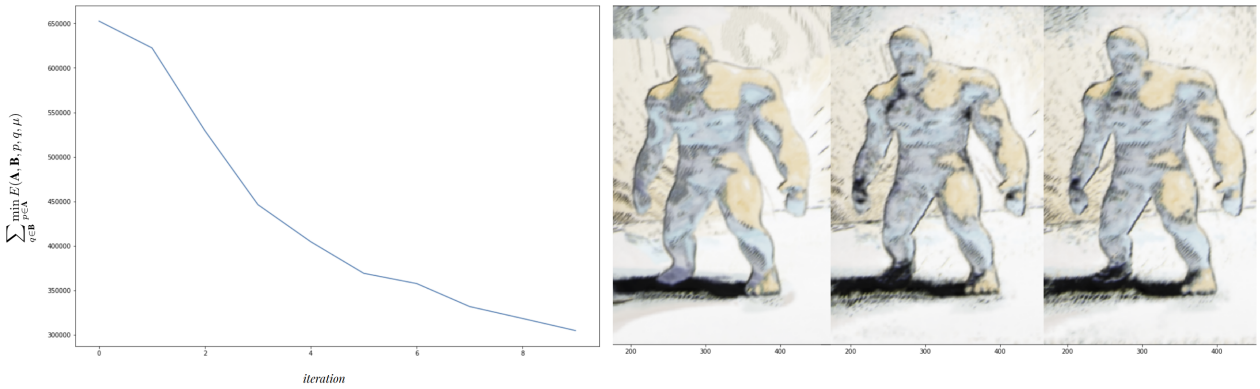


Figure 12: The final strategy with the erroneous filtering using the statistical approach proposed, on the LEFT example: direct NNF, CENTER: reversed NNF, RIGHT: reversed NNF with erroneous assignments filtering. At each iteration the filtering unmasks most feasible assignment leading to a lower global energy.

0.4.2 Summary and installation

The Stylit strategy has been implemented but results are not as good as the original in terms of quality. The PATCH MATCH algorithm is the most expensive step of all the algorithms it has been implemented in c++ and connected to the python environment via pybind11.

The project has a list of dependencies listed in the *requirements.txt* file once all packages are installed run make to compile the python packages needed to run the interface implemented in the notebook *Stylit Interface*. Source is available here: https://github.com/marioviti/OpenStylit.

Bibliography

[BSFG09] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[FJL+16] Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. StyLit: Illumination-guided example-based stylization of 3d renderings. *ACM Transactions on Graphics*, 35(4), 2016.

[JFA+15] Ondřej Jamriška, Jakub Fišer, Paul Asente, Jingwan Lu, Eli Shechtman, and Daniel Sýkora. LazyFluids: Appearance transfer for fluid animations. *ACM Transactions on Graphics*, 34(4), 2015.

[RT06] Guodong Rong and Tiow-Seng Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 109–116, 2006.